BioE/MCB/PMB C146/C246: Topics in Computational Biology and Genomics

Lecture 4) Scoring Alignments; Dynamic Programming

30 Jan 2003

Scribe: Ali Bashir *Reader*: Lee Chae

GLOBAL ALIGNMENT:

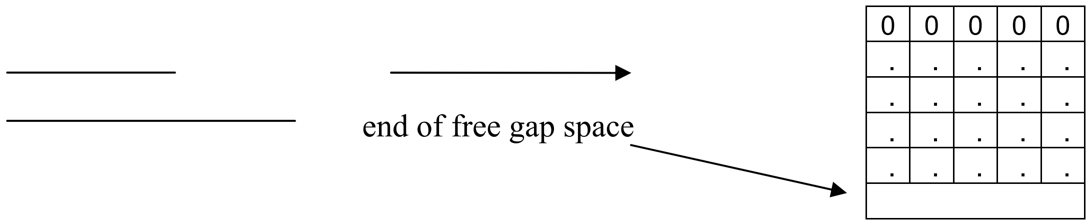
Use when sequence lengths are similar (n=m) – see example (Needlemen_Wunsch)

SEMIGLOBAL/LOCAL-GLOBAL ALIGNMENT

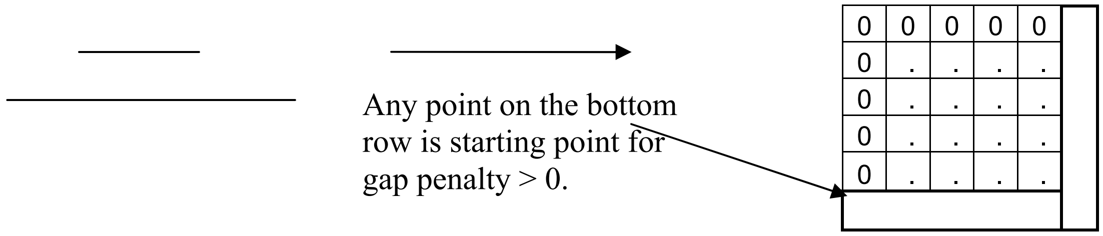
Method for handling sequences of unequal length ($n \neq m$)

Examples:

1.)

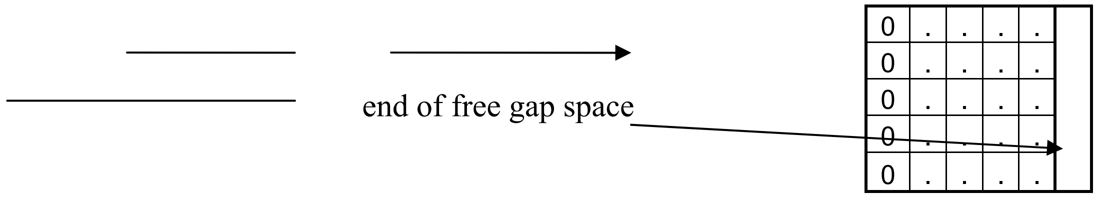


2.)


Any point on the bottom row is starting point for gap penalty > 0.
0 0 0 0 0
0
0
0
0

combination of 1 and 3

3.)

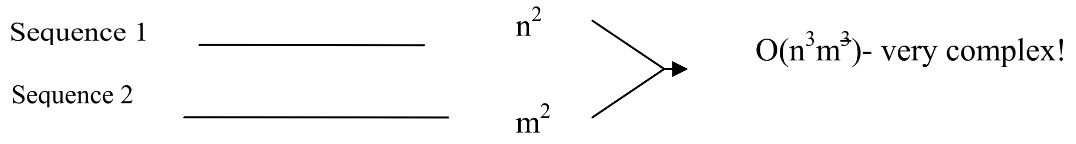


Local Alignment

Finding motifs; conserved subregions (Smith-Waterman)

Background

Using a naïve approach, all substrings would need to be compared against each other to find the best alignment. Such an algorithm takes $O(n^6)$.



We do not need to try all substrings, but all starting at different places. Using Needleman-Wunsch matrix to look at place where best score in middle is denoted where the tails of optimal substrings and traceback- $O(n^2 m^2) \sim O(n^4)$.

Two groups Goad and Kanehisa and Smith Waterson, revised previous dynamic programming algorithms by adding a zero state (the Goad and Kanehisa algorithm was actually more generalized and more complex, however Smith Waterson is more well-known).

Smith Waterman

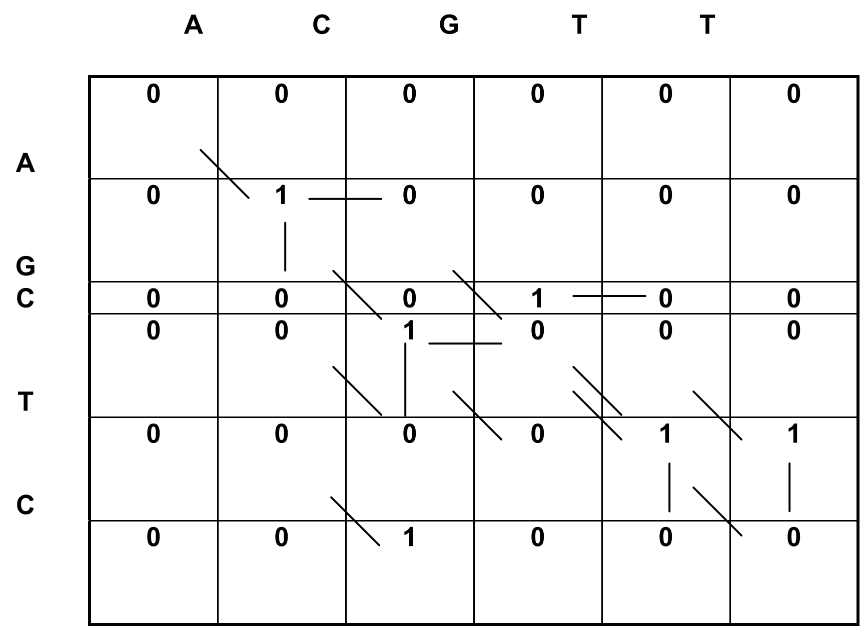
$$S_{ij} = \max \begin{cases} S_{i-1,j-1}+s_{ij} & \text{(i.e. extend the alignment without adding a gap)} \\ S_{i-1,j}-g & \text{(extend the alignment by adding a gap to the second sequence)} \\ S_{i,j-1}-g & \text{(add a gap to the first sequence)} \\ 0 & \text{(start the alignment over)} \end{cases}$$

$S_{0,0} = 0$

To score:

Look for highest score anywhere in matrix and trace the alignment until you reach a zero.

Example:



		A	C	G	T	T
	0	0	0	0	0	0
A	0	1	0	0	0	0
G	0	0	0	1	0	0
C	0	0	1	0	0	0
T	0	0	0	0	1	1
C	0	0	1	0	0	0

Look for highest score anywhere in the matrix, traceback from there.

Some optimal alignments:

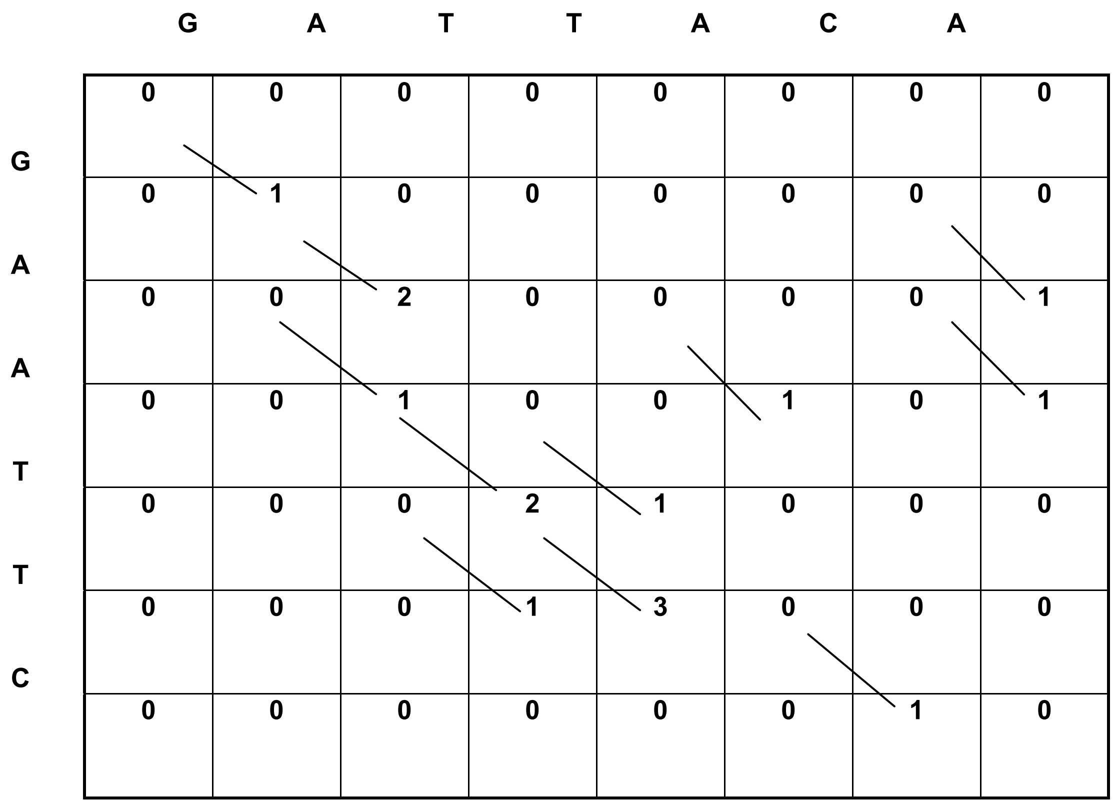
ACGTT,	A- C,	A,	C,
A- GCT,	AGC,	A,	C,

Finding longest common subsequence

Essentially you are looking for the longest path in the matrix, thus every time a path ends, you must start over, ie. only increment for matches, otherwise set the matrix value to 0.

$$S_{ij} = \max \begin{cases} S_{i-1,j-1} + 1 \text{ (if } S_{ij} \text{ is a match)} \\ 0 \text{ (if mismatch)} \end{cases}$$

Score by looking for highest score and tracing back to zero.



		G	A	T	T	A	C	A
	0	0	0	0	0	0	0	0
G	0	1	0	0	0	0	0	0
A	0	0	2	0	0	0	0	1
A	0	0	1	0	0	1	0	1
T	0	0	0	2	1	0	0	0
T	0	0	0	1	3	0	0	0
C	0	0	0	0	0	0	1	0

Longest substring:

Length 3, from matrix, substring is ATT

LOCAL VS. GLOBAL

Local

Useful beyond the cases mentioned above, also because evolution will mutate everything but what is important. Portions of proteins can be partially homologous.
S_{all} α log(length^2)

Global

S_{all} α length

DISTANCE

Distance vs. Similarity

So far we have been using Similarity, now let us introduce distance and compare.

Rules	Distance?	Similarity?
D(a,b) = 0 if a =b	Y	N
D(a,b) = D(b,a)	Y	Y
$D(a,b) \leq D(a,c) + D(c,b)$	Y	N
D(a,a) = D(b,b)	Y	N
Local alignment	N	Y
Phylogeny Dist. Methods	Y	N

Hamming Distance

An effective method, however it doesn't include the distance needed to accommodate gaps.

AGTC
ATTA $\longrightarrow$ Distance = number of mismatches = 2

Levenshtein (or Edit) Distance

Levenshtein distance helps account for the gapping inadequacy of Hamming distance.

AG- TCC
CGCTCA $\longrightarrow$ Make as many edits (change residue or insert gap) you need to convert one sequence to another (changes not made in both sequences)

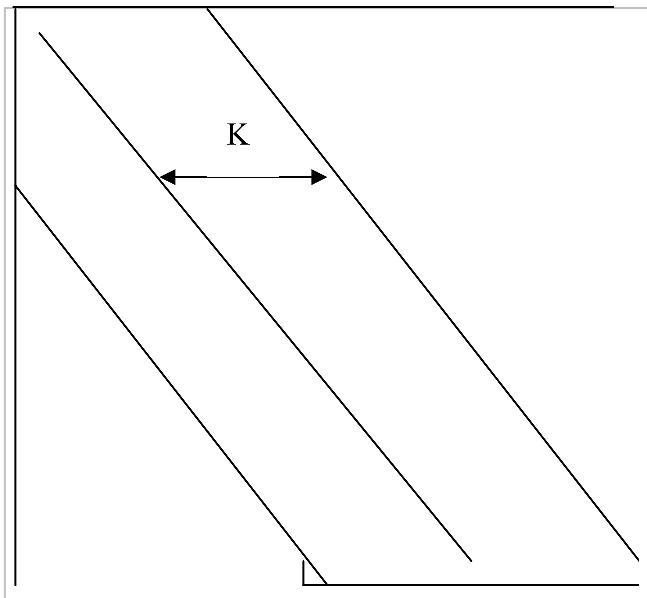
Smith-Waterman - showed how to convert between similarity and distance

Similarity $\longleftrightarrow$ Distance Conversion

$d_{ij} = S_{max} - S_{ij}$	highest score of residue
$d_k = (S_{max} - k)/2 + p(k)$	where k = gap length
$D(a,b) = (S_{max}(n+m)) / 2 - \underline{S}(a,b)$	overall distance

Special Case: Band Alignment

If you already know that the sequences being examined are very similar to one another, than one can presume that they will hold close to the diagonal in a matrix (this method only holds true for global alignment).



Deviation from center line is insertion of gap, and:

Σ gaps < 2k + 1] (k = bandwidth)
$S0,0 = 0$
$Si,o = ip$
$So,j = jp$

IB: Inside Band ($-k \leq i-j \leq k$)

$$S_{ij} \overset{(IB)}{\text{----max}} \begin{cases} S_{i-1,j-1} + S_{ij} \\ S_{i-1,j} + P, \text{ if } IB(S_{i-1,j}) \\ S_{i,j-1} + P, \text{ if } IB(S_{i,j-1}) \end{cases}$$

Must check to make sure that these still fall within band.

Running time of this algorithm is approximately O(kn). It is critical to develop a methodology at score at end, and then see if it is close to optimal. If the score is not optimal, than rerun the algorithm.

$$\text{Best score:} \quad \overbrace{M(n-k+1)}^{\text{all other positions, assuming all match perfectly}} + \overbrace{2(k+1)P}^{\text{\# gaps multiplied by gap penalty}}$$

M = match score

P = gap penalty

N = length of sequence